

Егор Хайруллин,
руководитель отдела
инфраструктуры рекомендательных систем

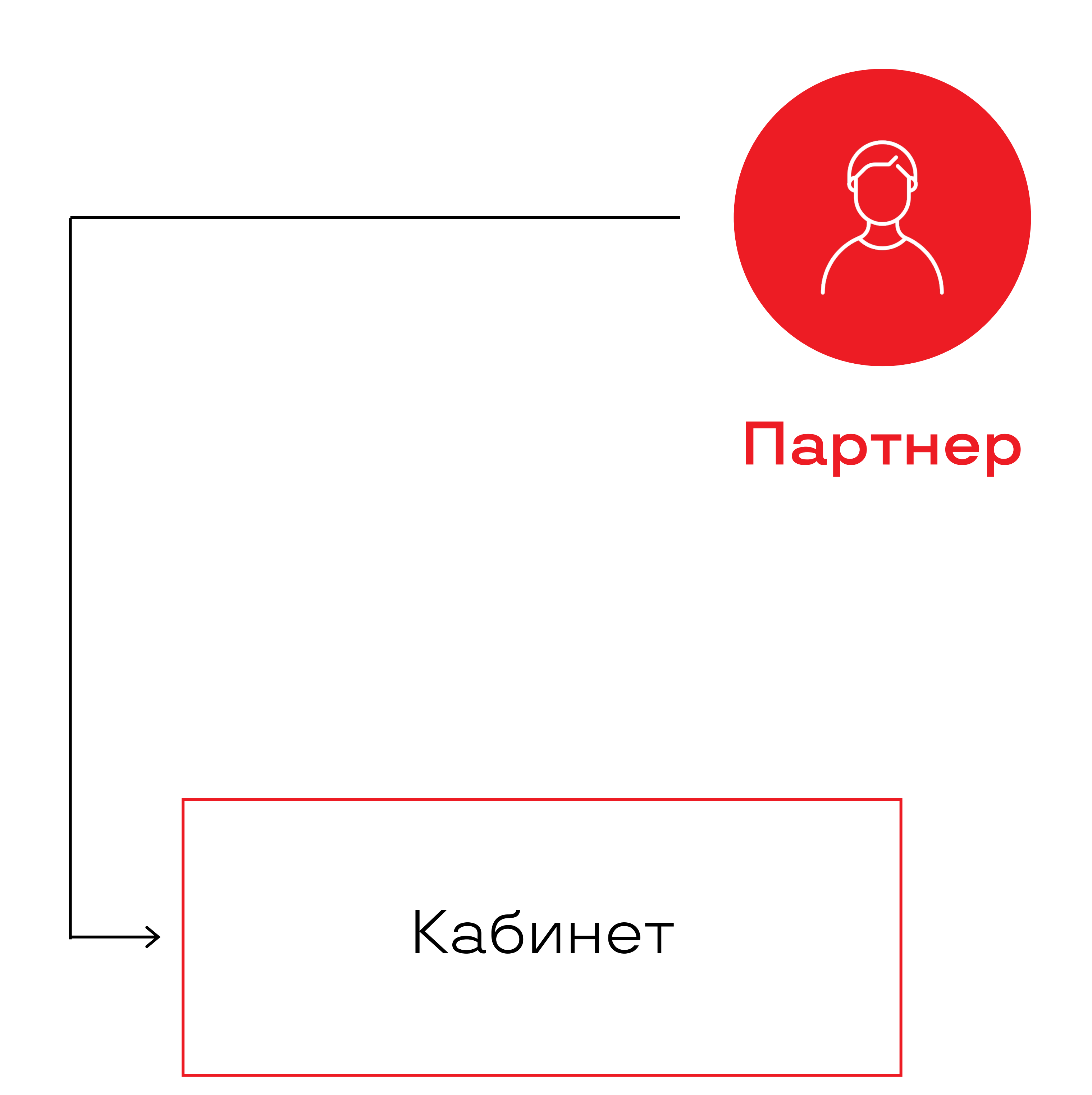


Про меня

+ 9+ лет работаю в Яндексе
+ Занимаюсь контент-системой рекламы
+ Преподаю в ШАДе
+ Люблю плюсы



Егор Хайруллин


Партнер
Кабинет

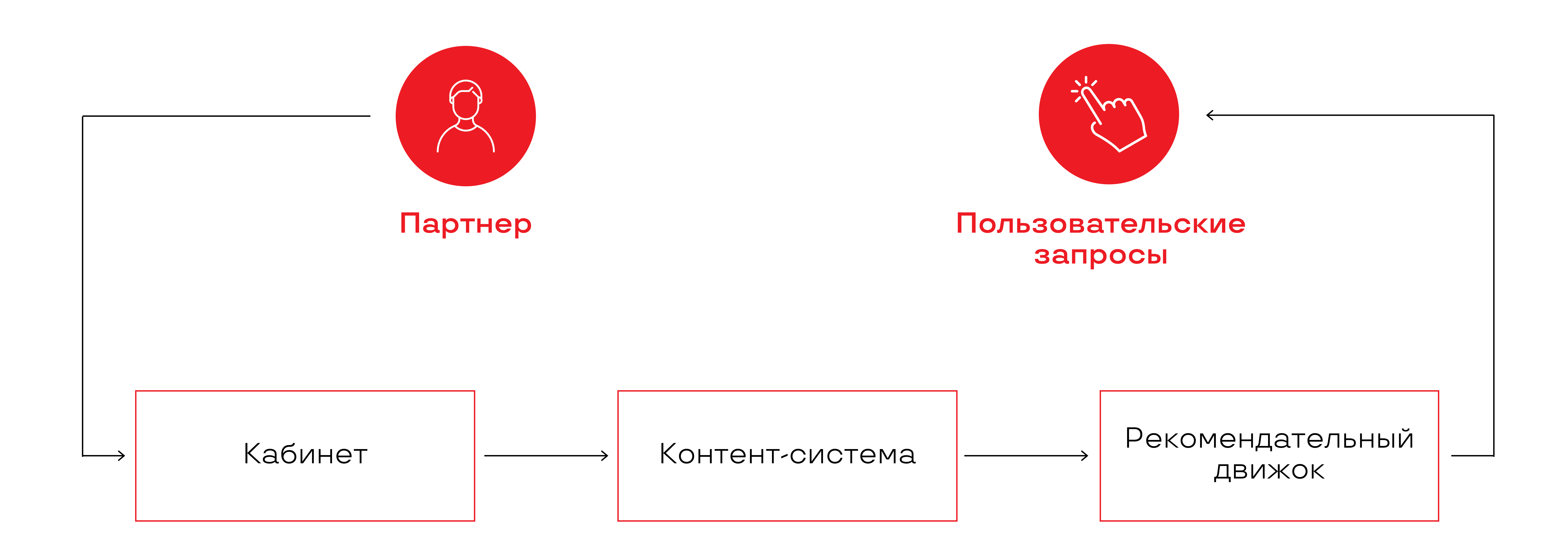




Кабинет

Рекомендательный движок

Контент-система собирает и подготавливает все данные



Что не так с Batch-подготовкой данных

Что не так с Batch-подготовкой данных

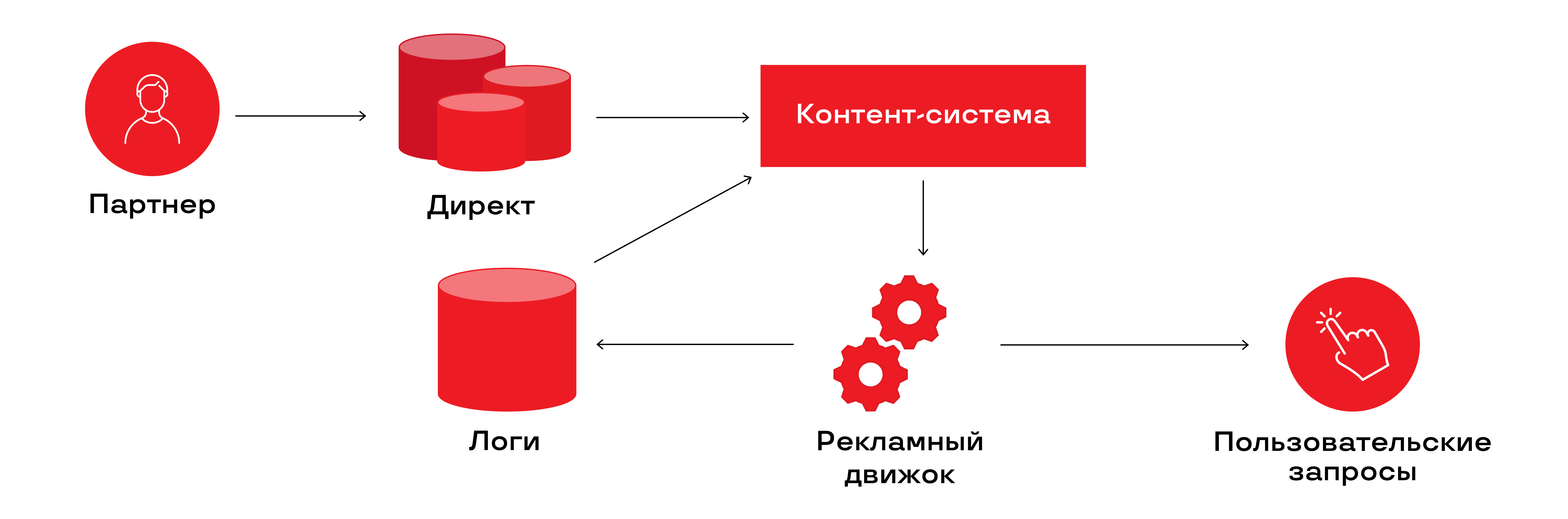
+ Большая задержка в обновлении данных
+ Очень большая стоимость в масштабировании

При этом в данных очень мало изменений, и большая часть подготовки идёт вхолостую

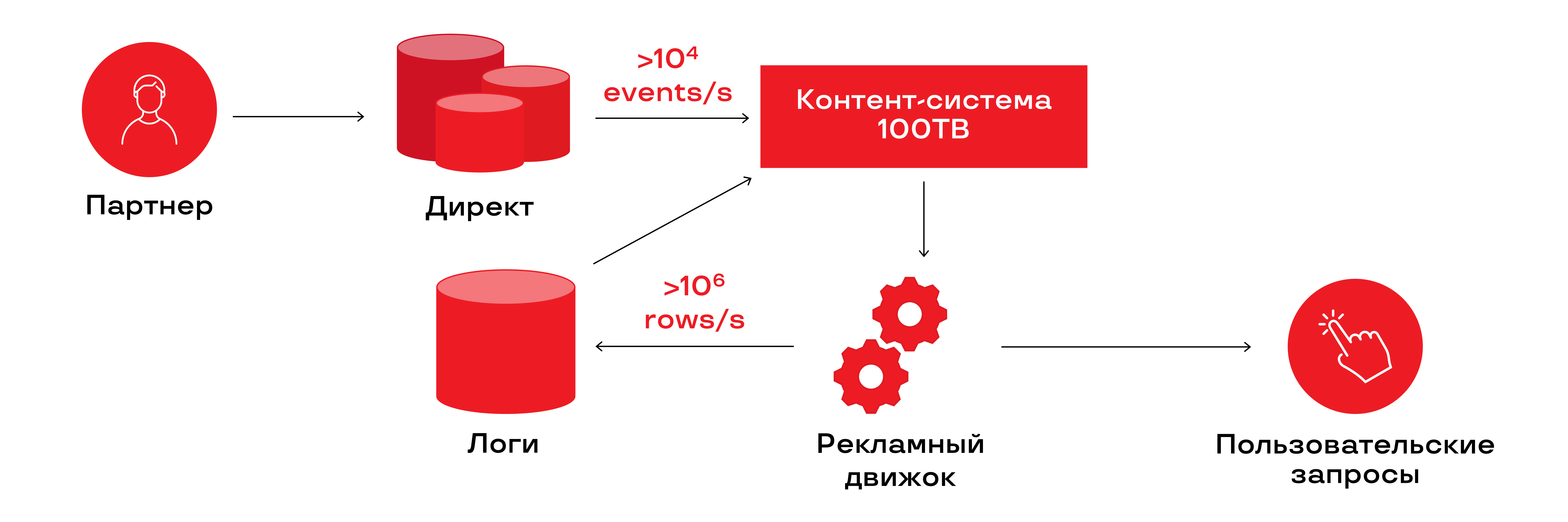
Нужна событийная система

+ Будем платить лишь за реальные изменения
+ Проще всего реализовать с помощью streaming processing

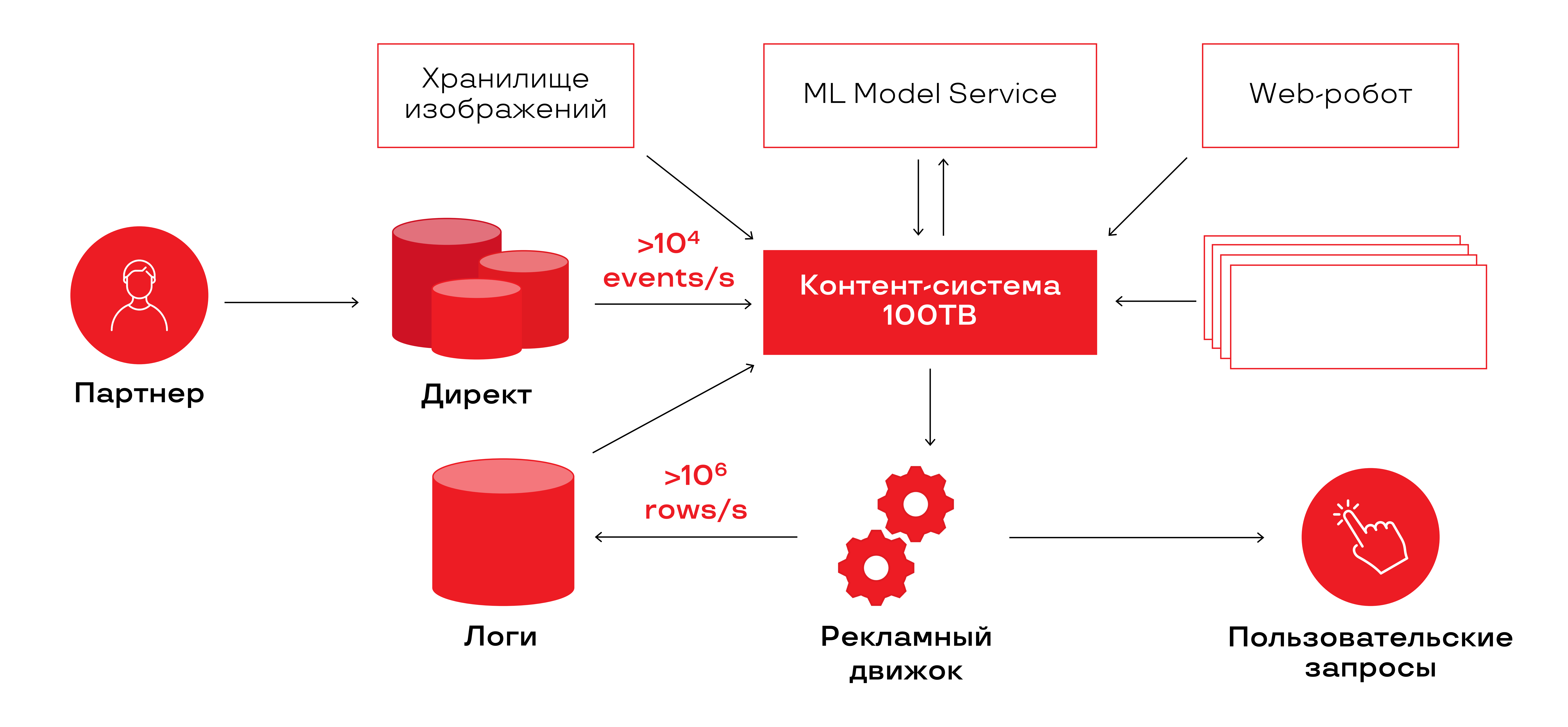
Контент-система собирает и подготавливает все данные Рекламы



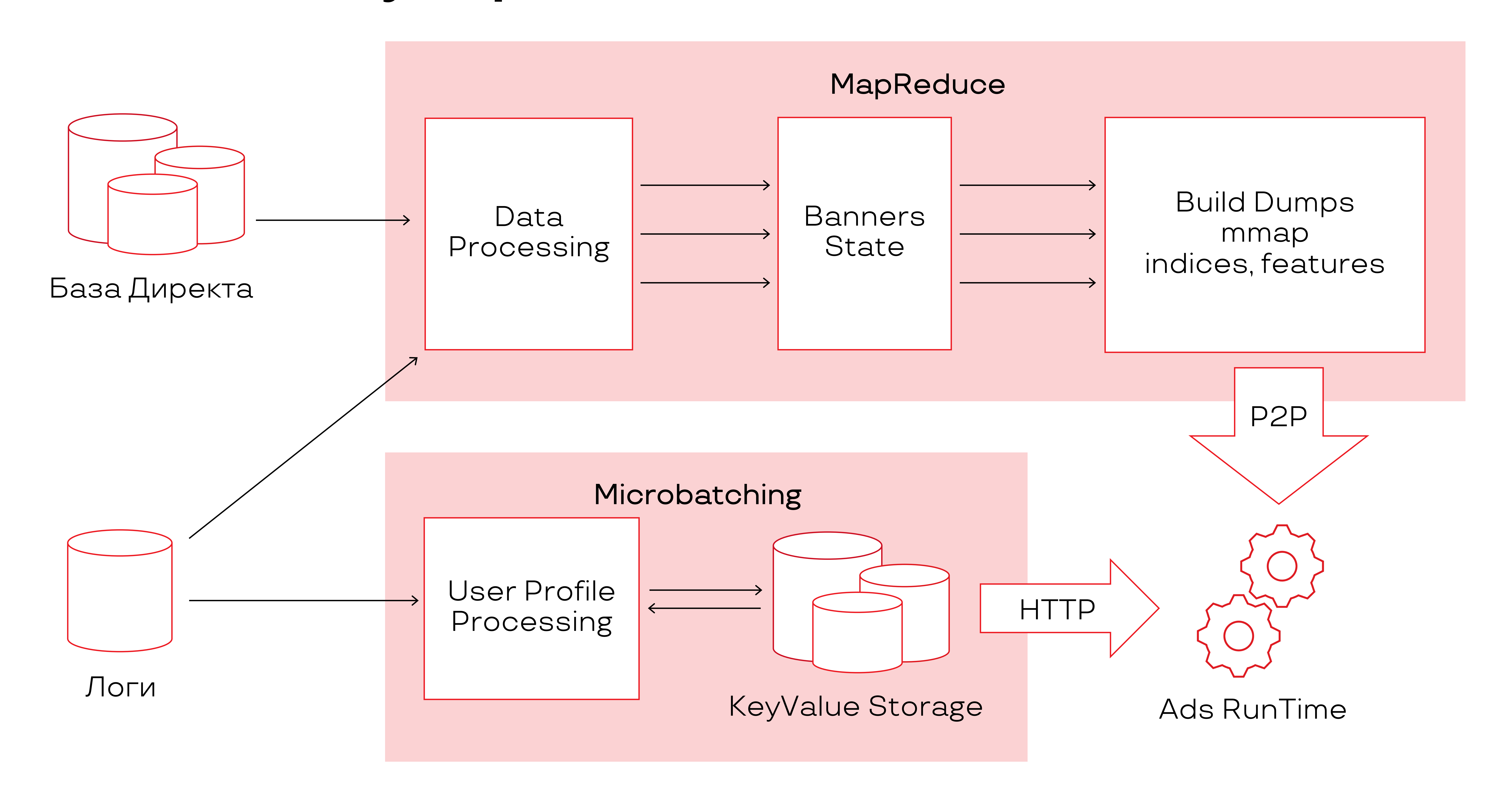
Контент-система — высоконагруженная система



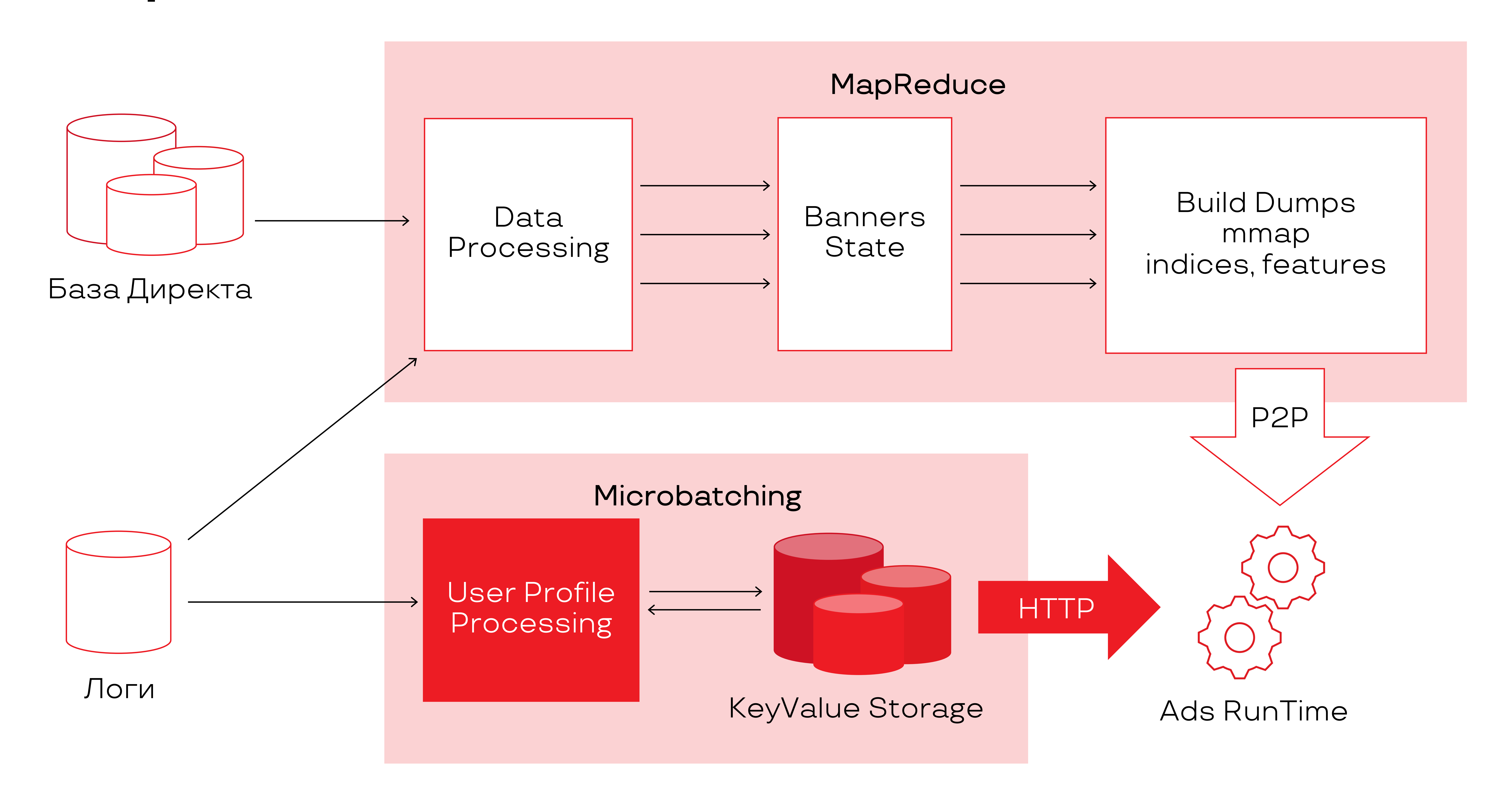
Контент-система — открытая система



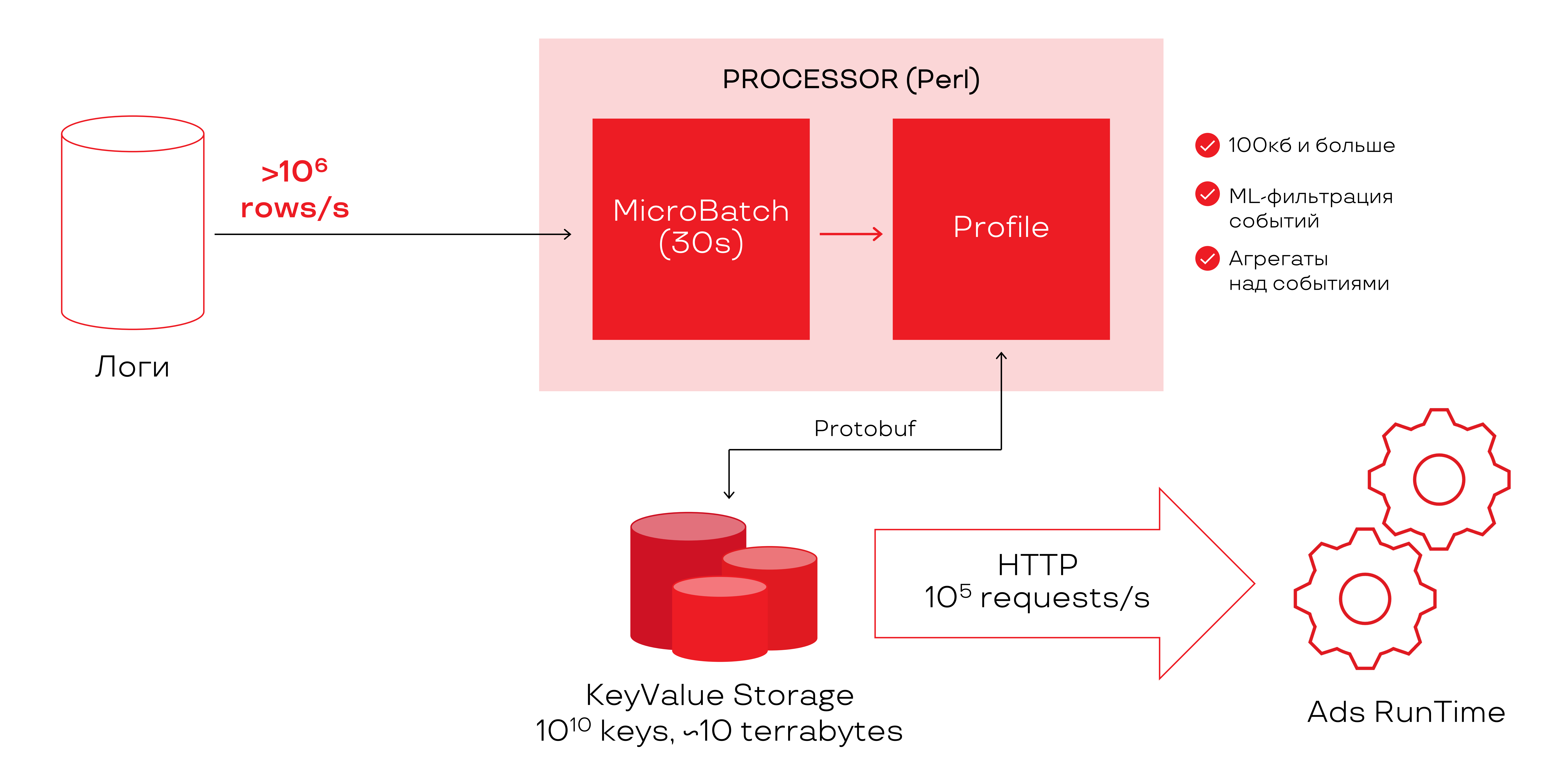
Как была устроена контент-система



Первый шаг – User Profile

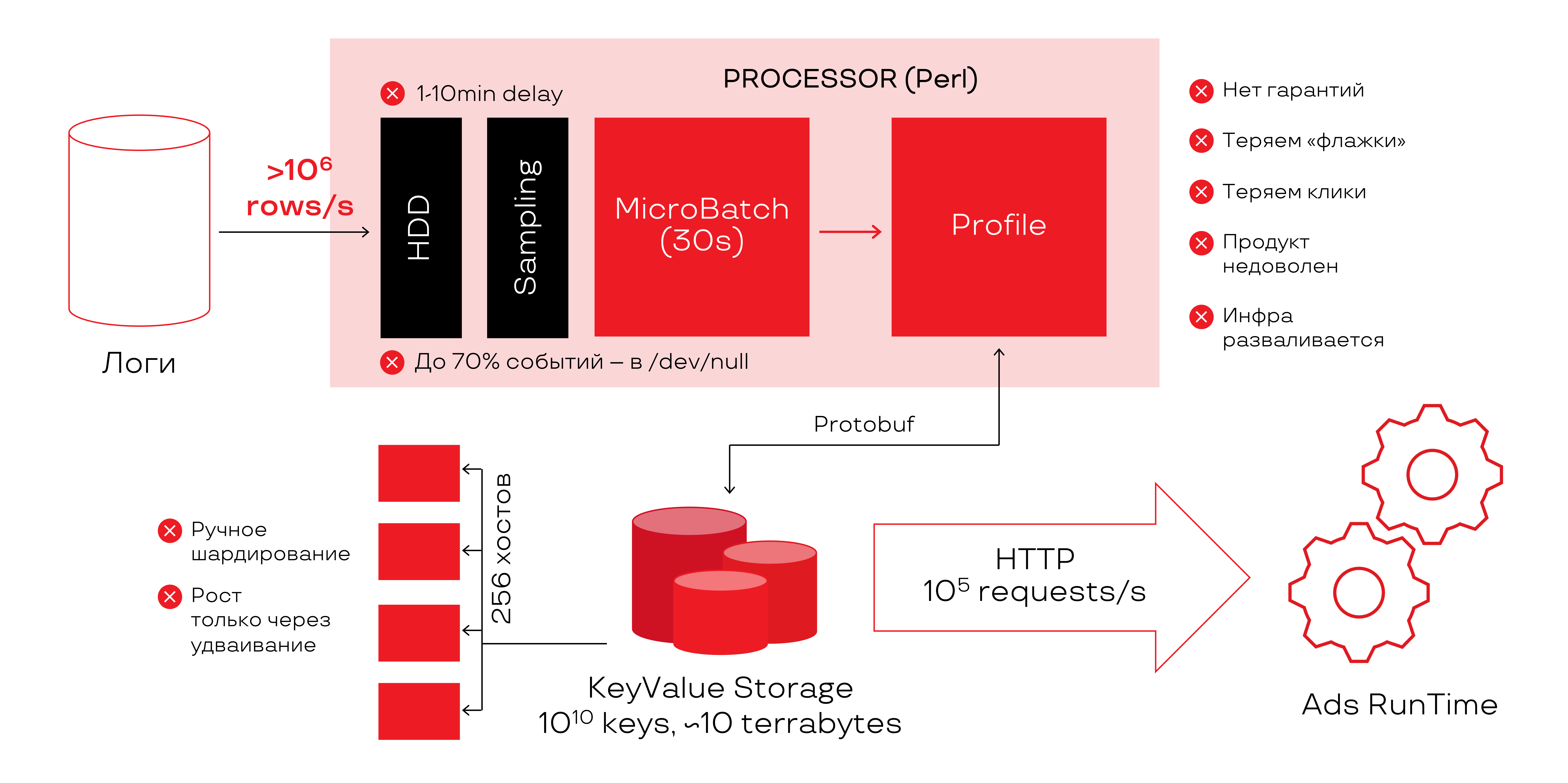


User Profile — уже был streaming-системой

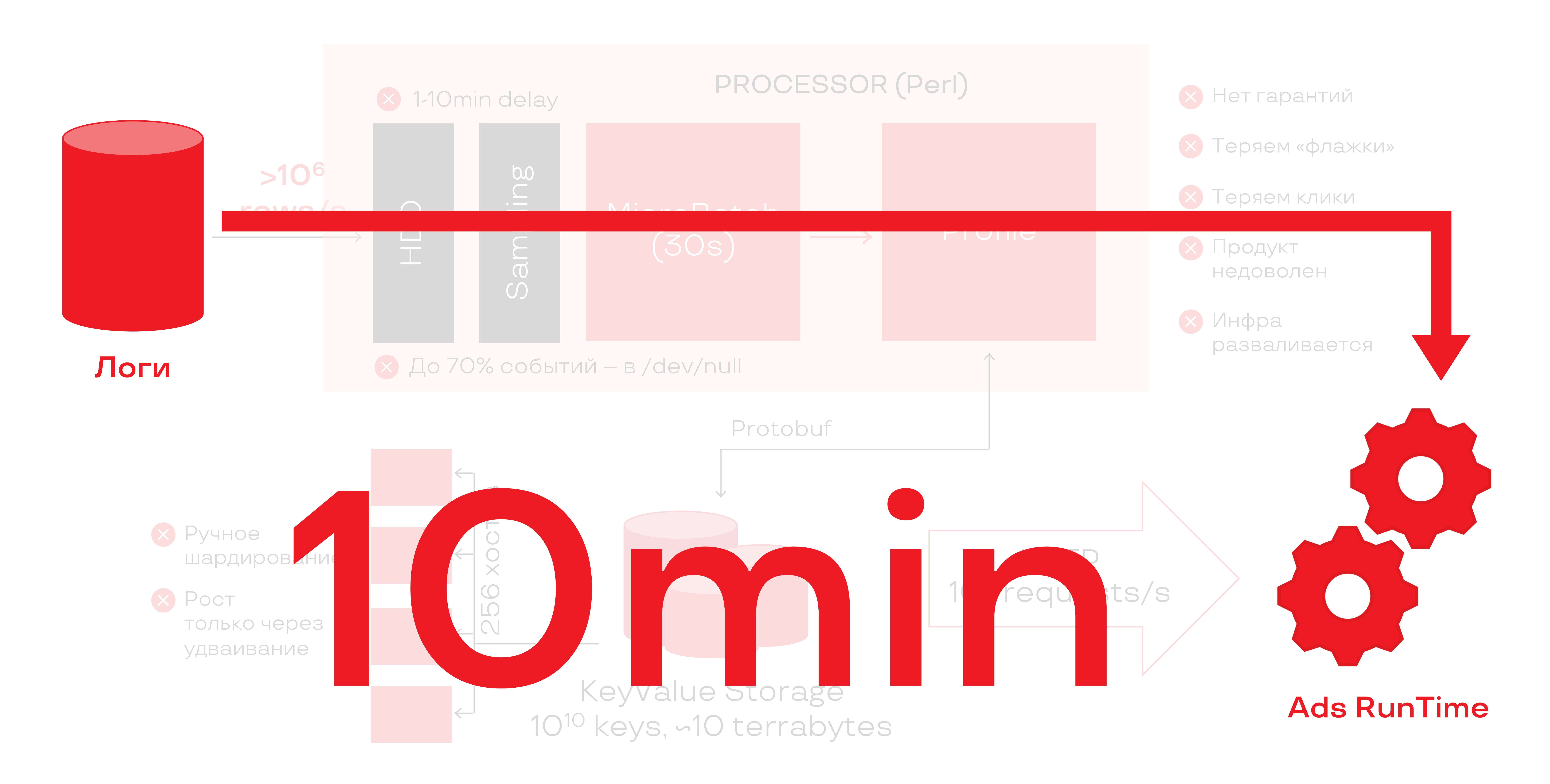


- 100кб и больше
- ML-фильтрация событий
- Агрегаты над событиями

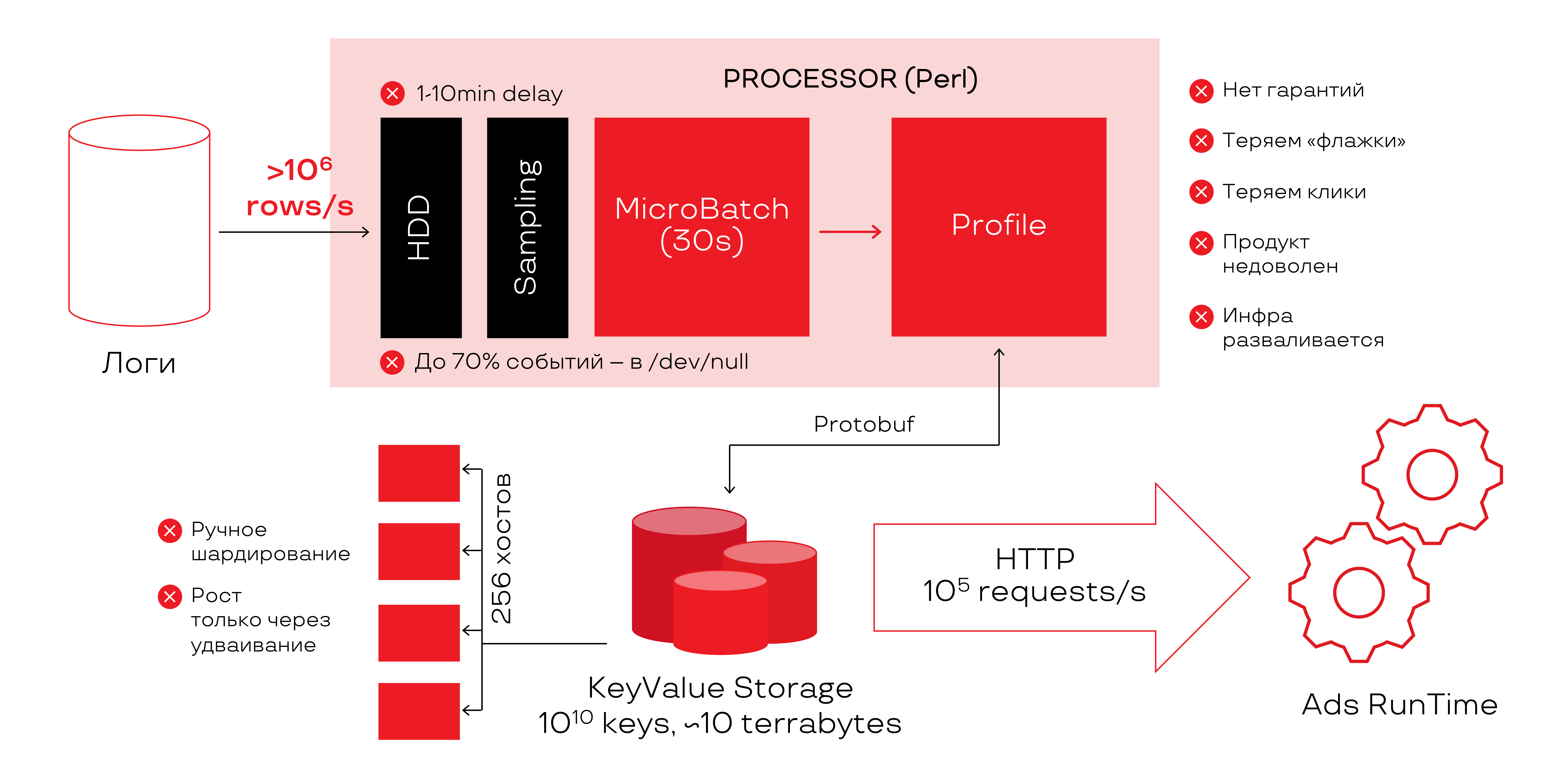
Но User Profile был плохой системой



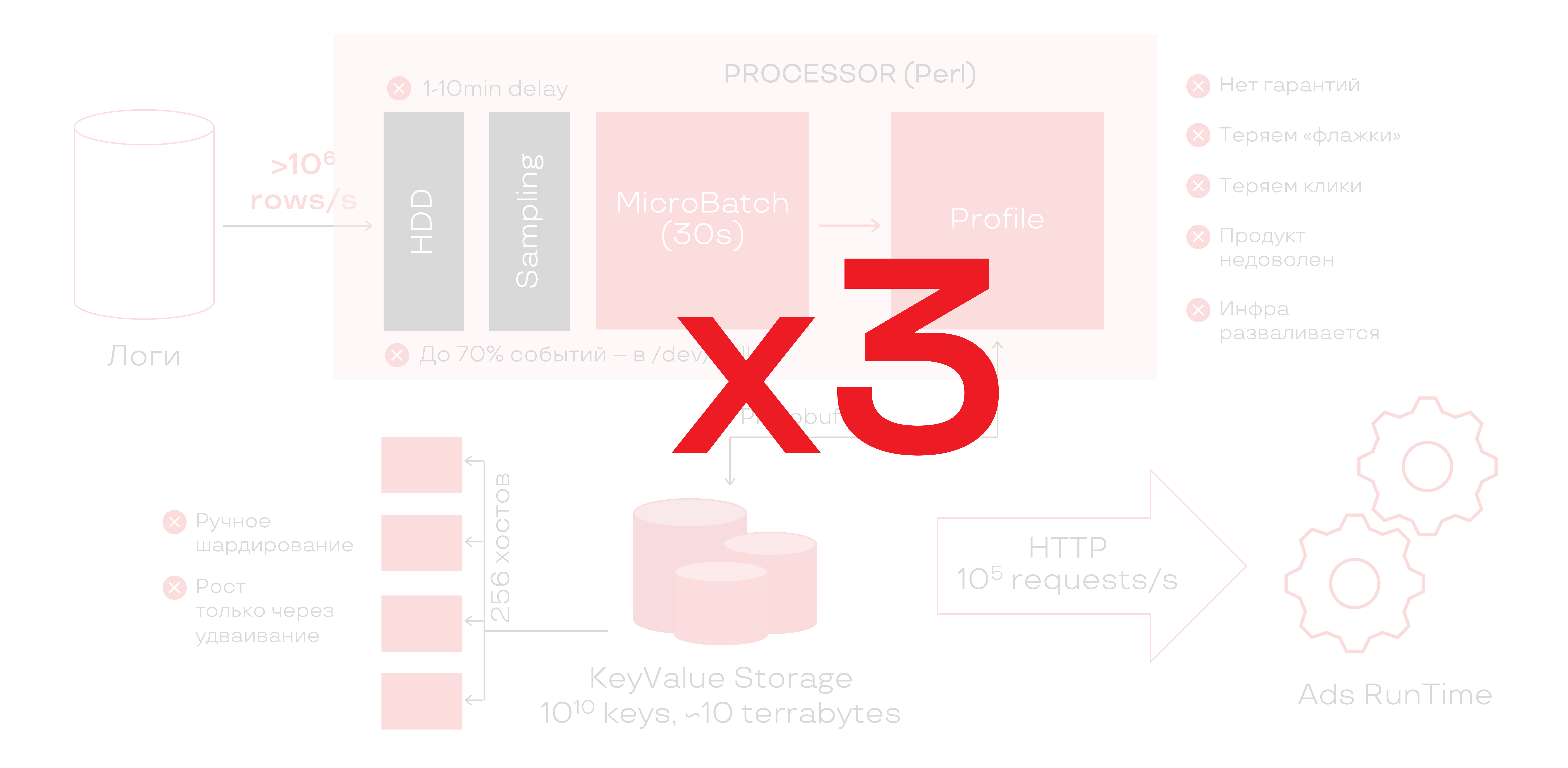
Но User Profile был плохой системой



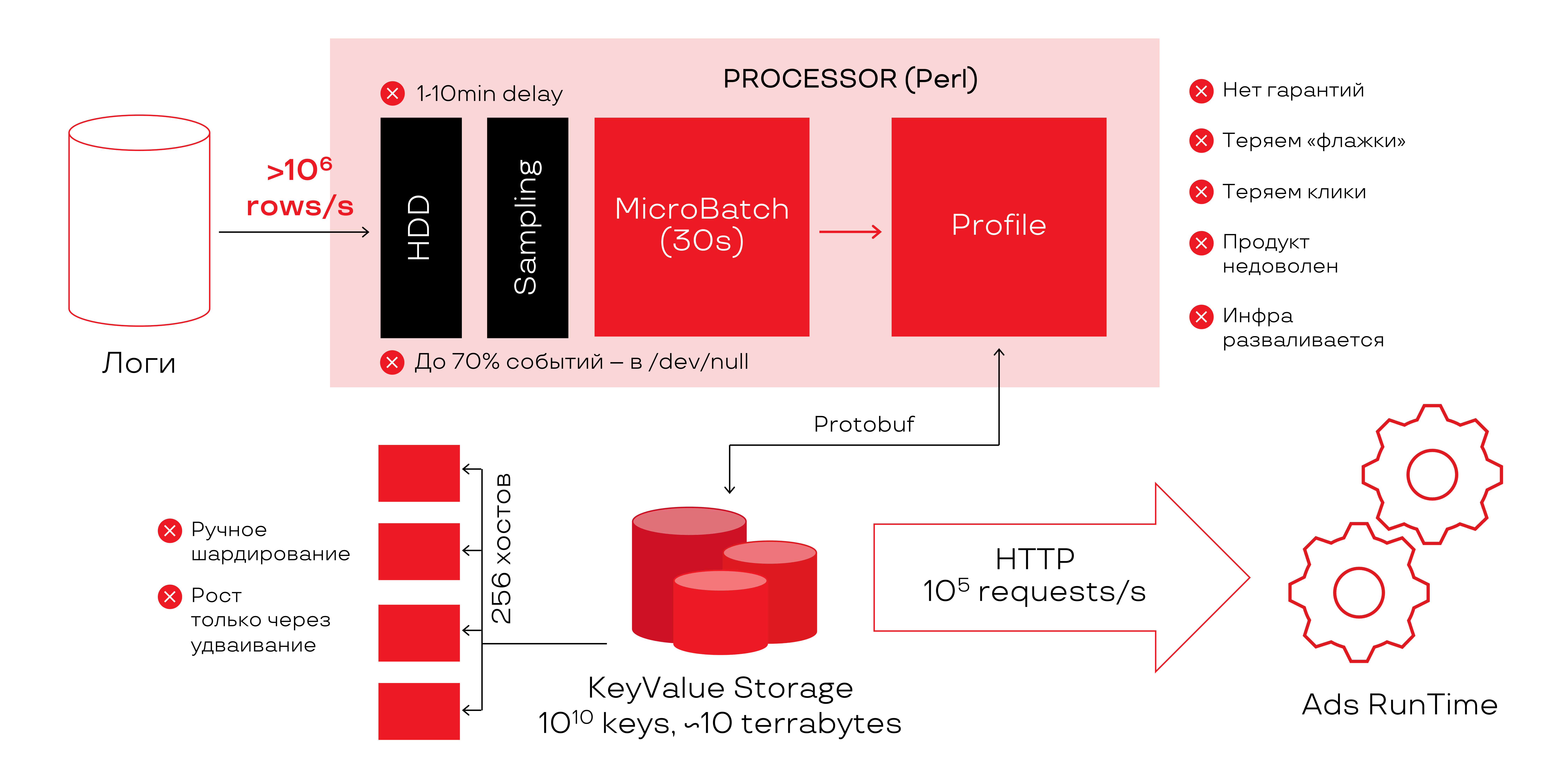
Но User Profile был плохой системой



Но User Profile был плохой системой



Но User Profile был плохой системой



Нам нужен классный Streaming User Profile!

Нужен хороший Storage

+ Распределенный Key-Value
+ Автошардируемость
+ Устойчивость к выпадению хостов или ДЦ
+ Транзакционность
+ Поддержка аналитических запросов
+ 1GB/s записи, 1M rows/s чтения

Выбрали YT DynTables

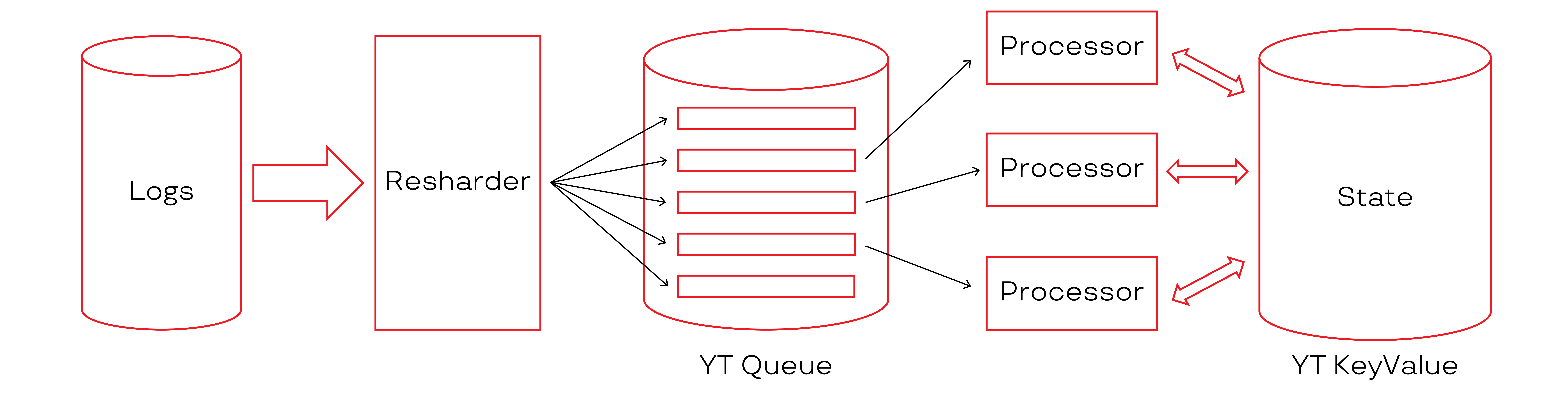
+ YT – проверенная система, давно используется для MapReduce
+ DynTables – реализация Key-Value внутри YT, диск+память
+ Полноценная поддержка MapReduce

А фреймворк?

+ Exactly-once – это просто
+ Хотим C++
+ Возможность Low-level-оптимизации

Решили писать BigRT

Схема типового BigRT-процессинга



Чтобы BigRT заработало эффективно, понадобились доработки

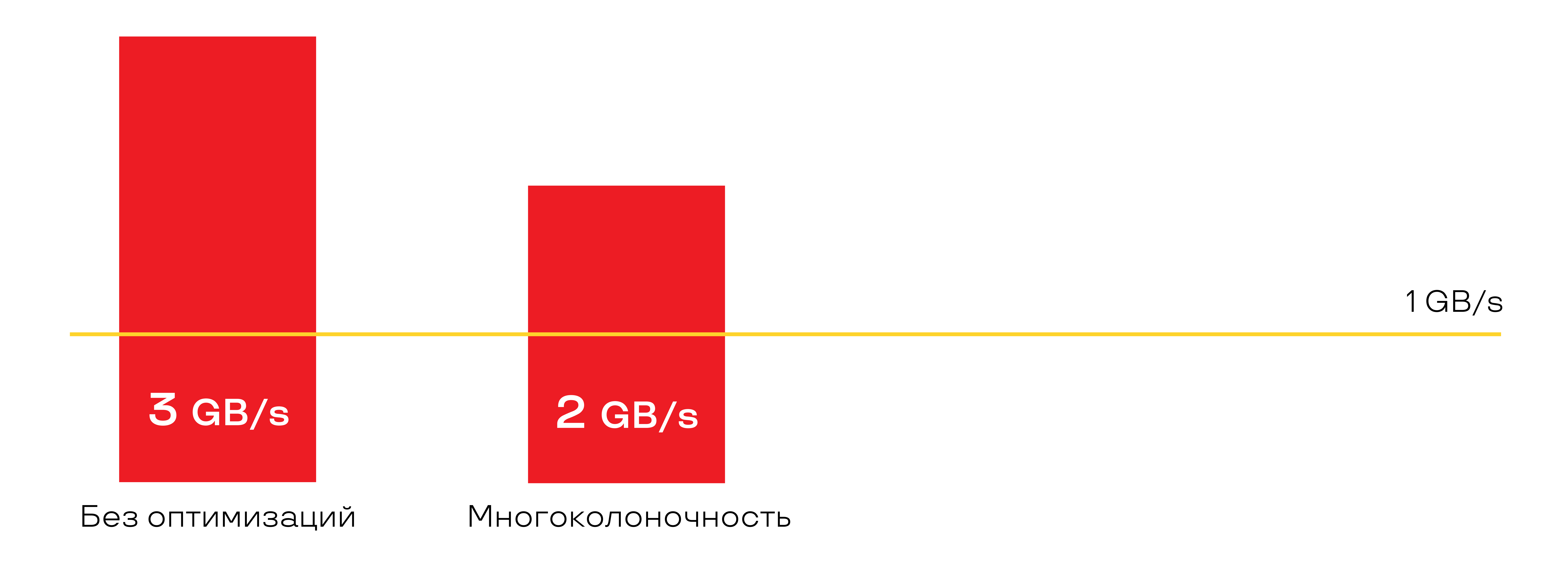
+ Балансировка шардов между воркерами
+ Сабшардирование для параллельной обработки шарда
+ Управление размером микробатчей
+ Кэширование профилей (с инвалидацией!)
+ Решардирование без материализации
+ Мониторинги и инструменты для управления
+ Удобство использования

Самая важная проблема – поток на запись

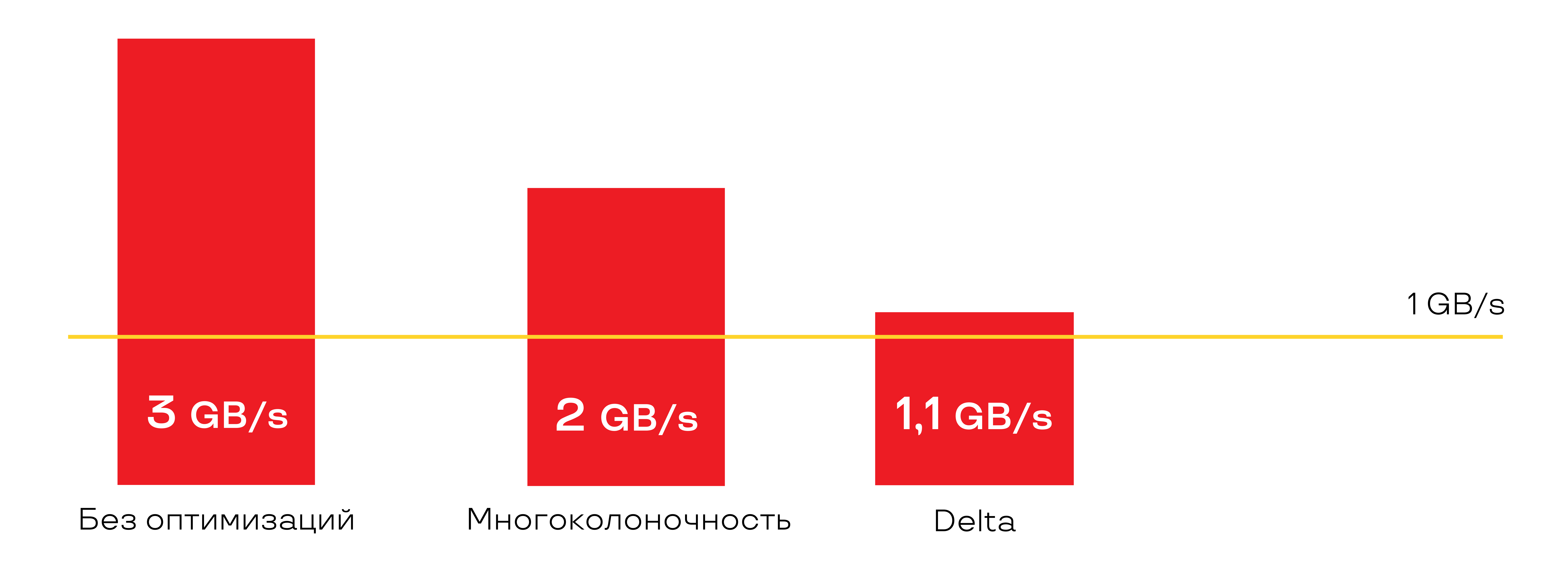
Самая важная проблема – поток на запись



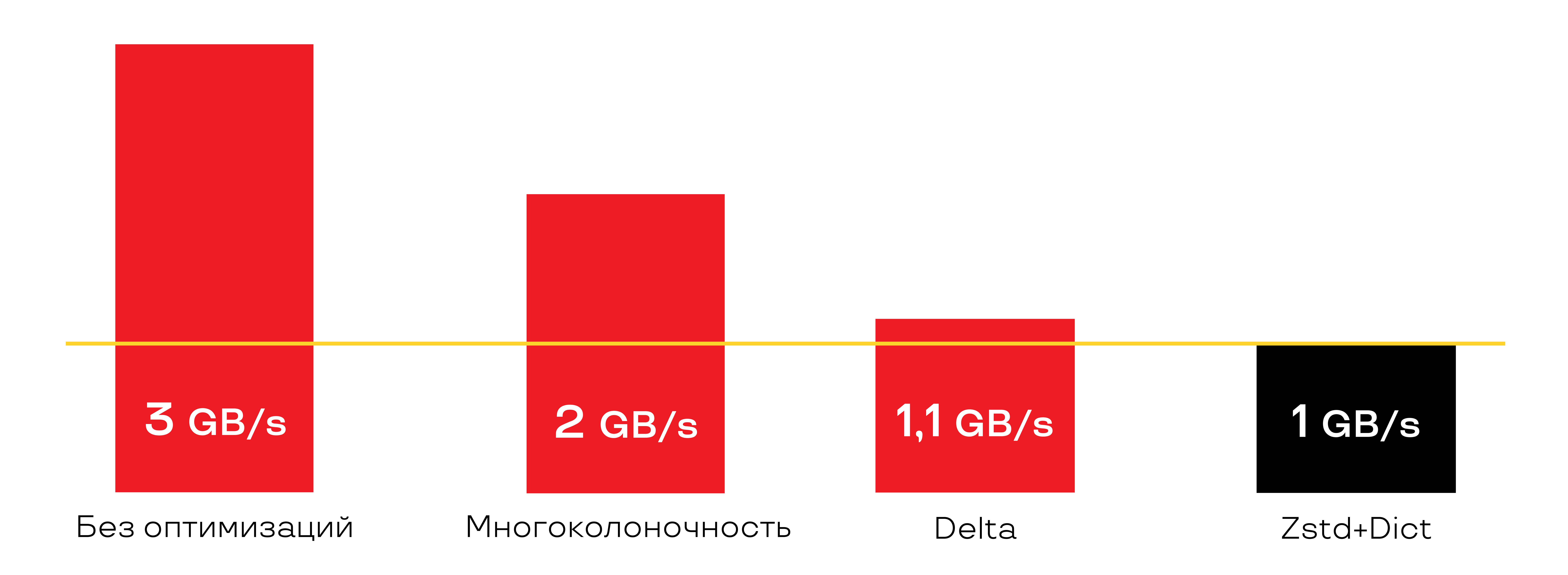
Самая важная проблема – поток на запись



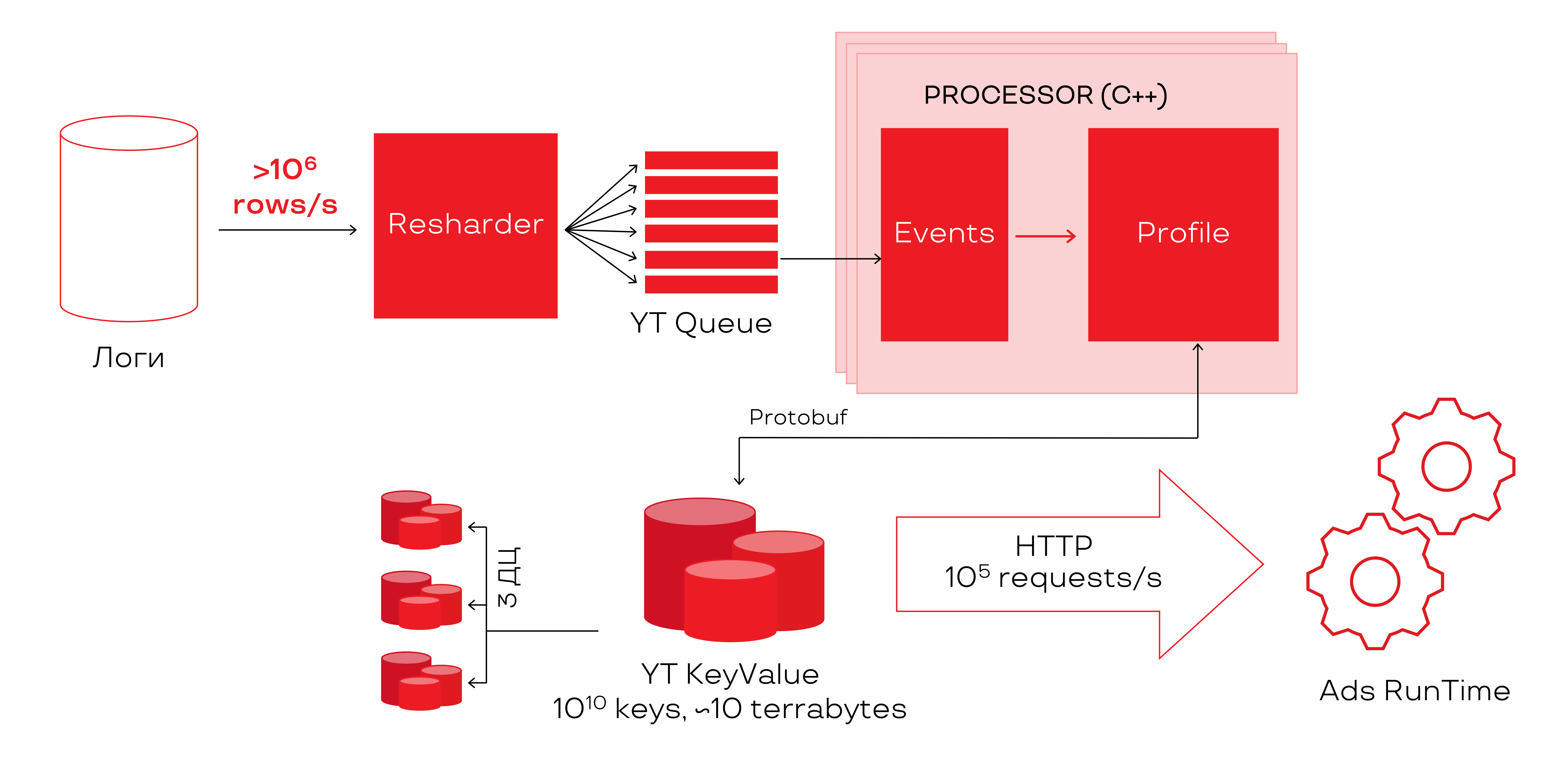
Самая важная проблема – поток на запись



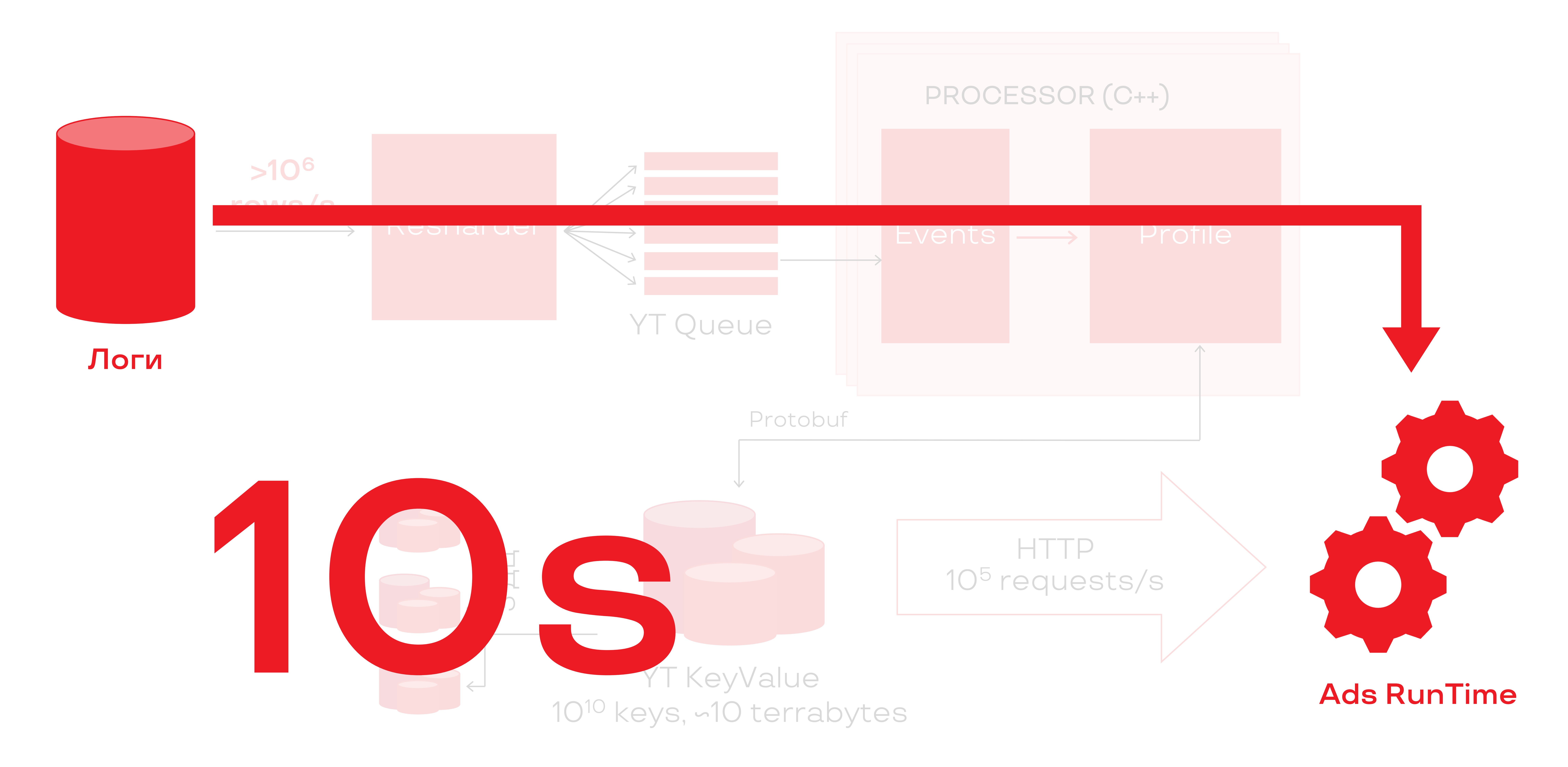
Самая важная проблема – поток на запись



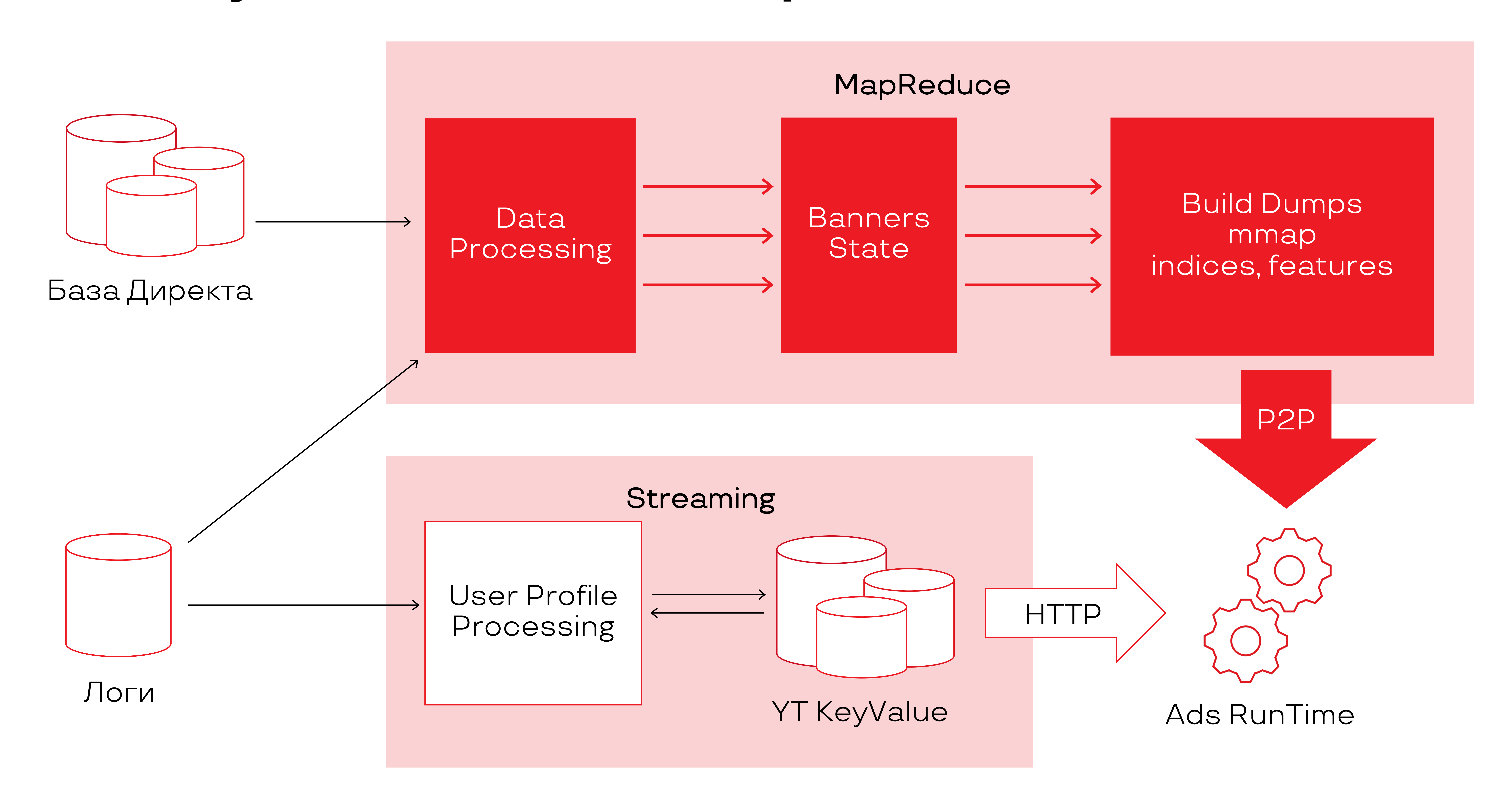
User Profile — стал хорошей системой



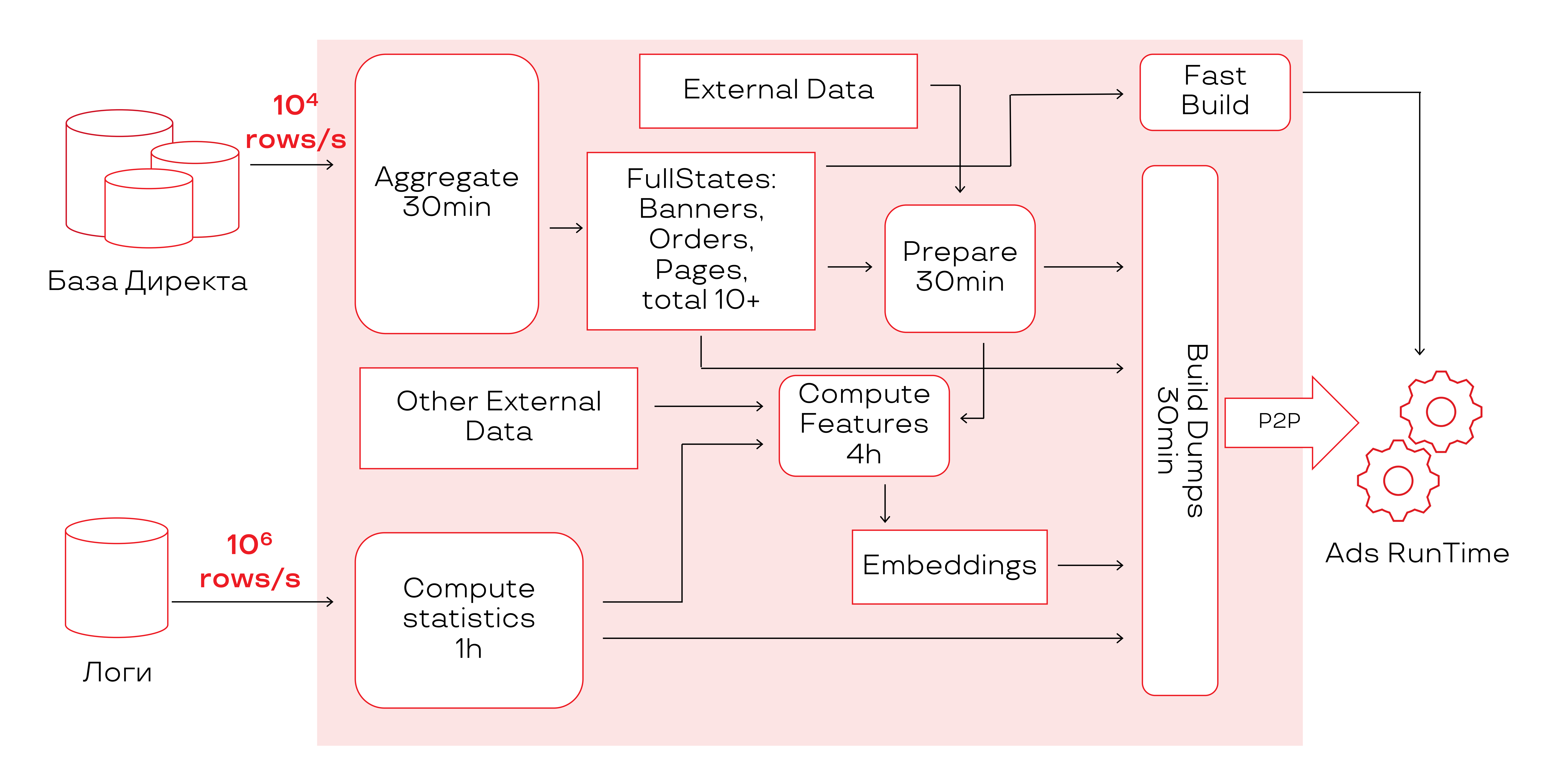
User Profile — стал хорошей системой



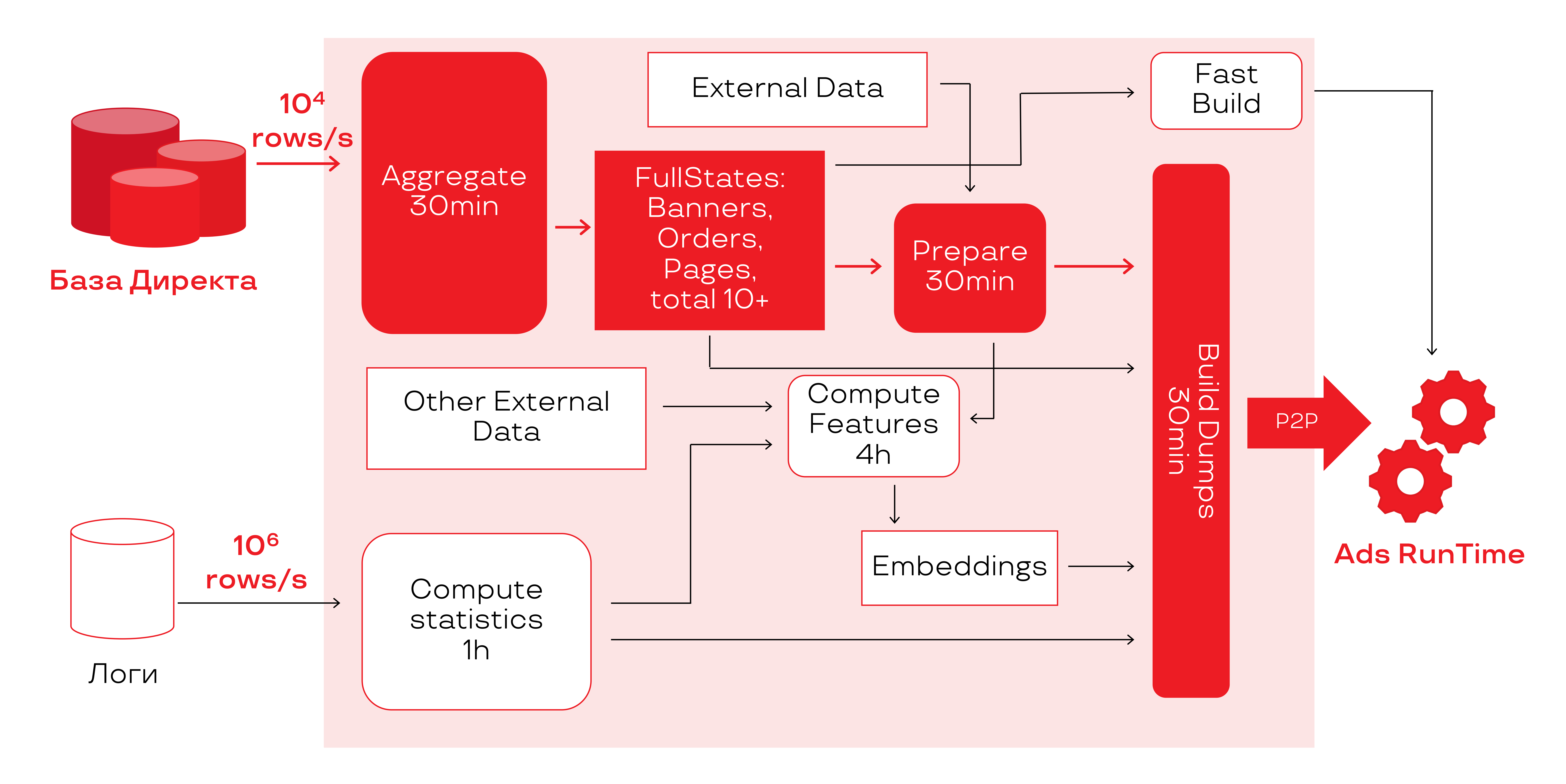
Следующий шаг – MapReduce-часть



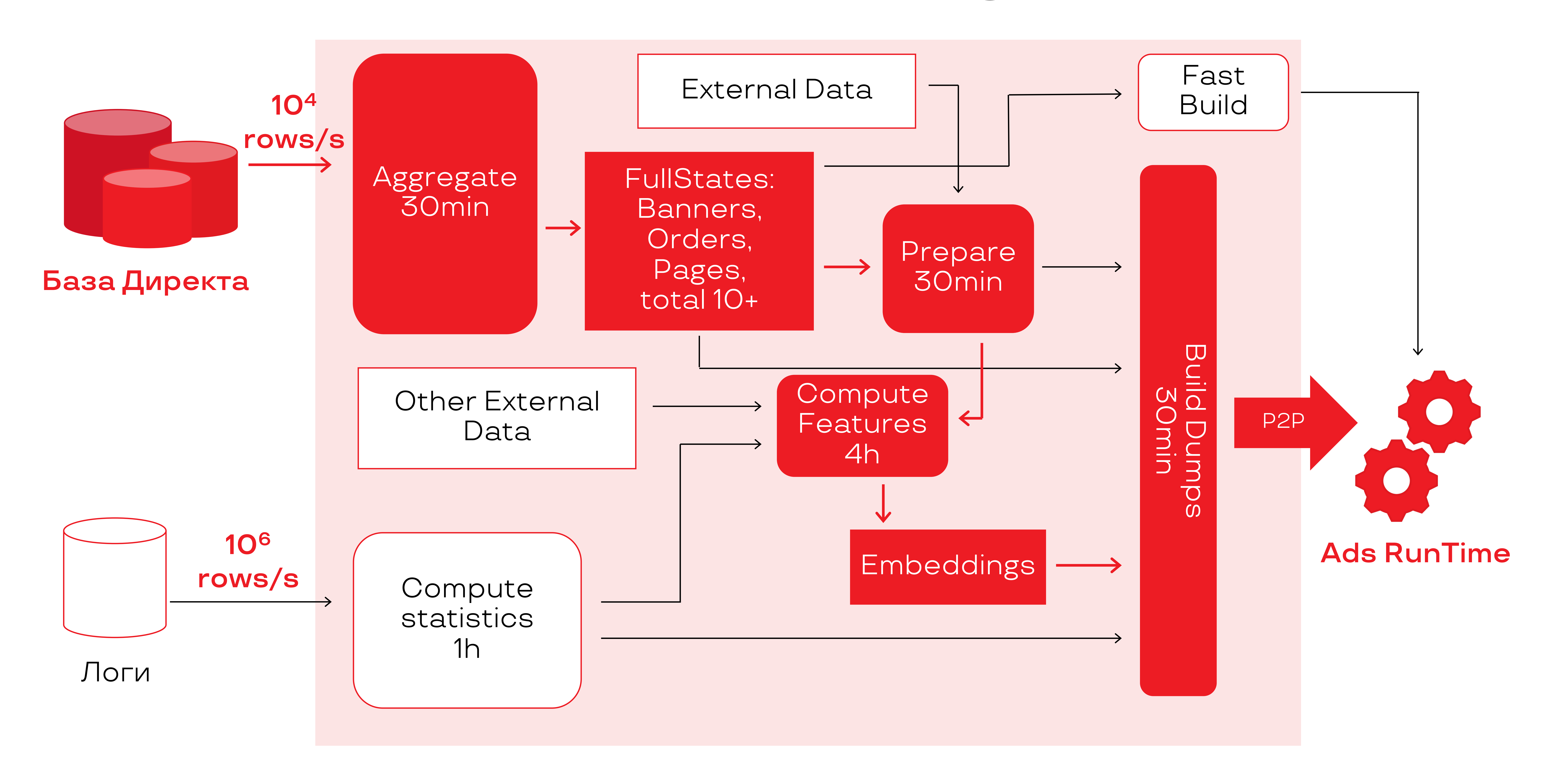
MapReduce – сложно и долго



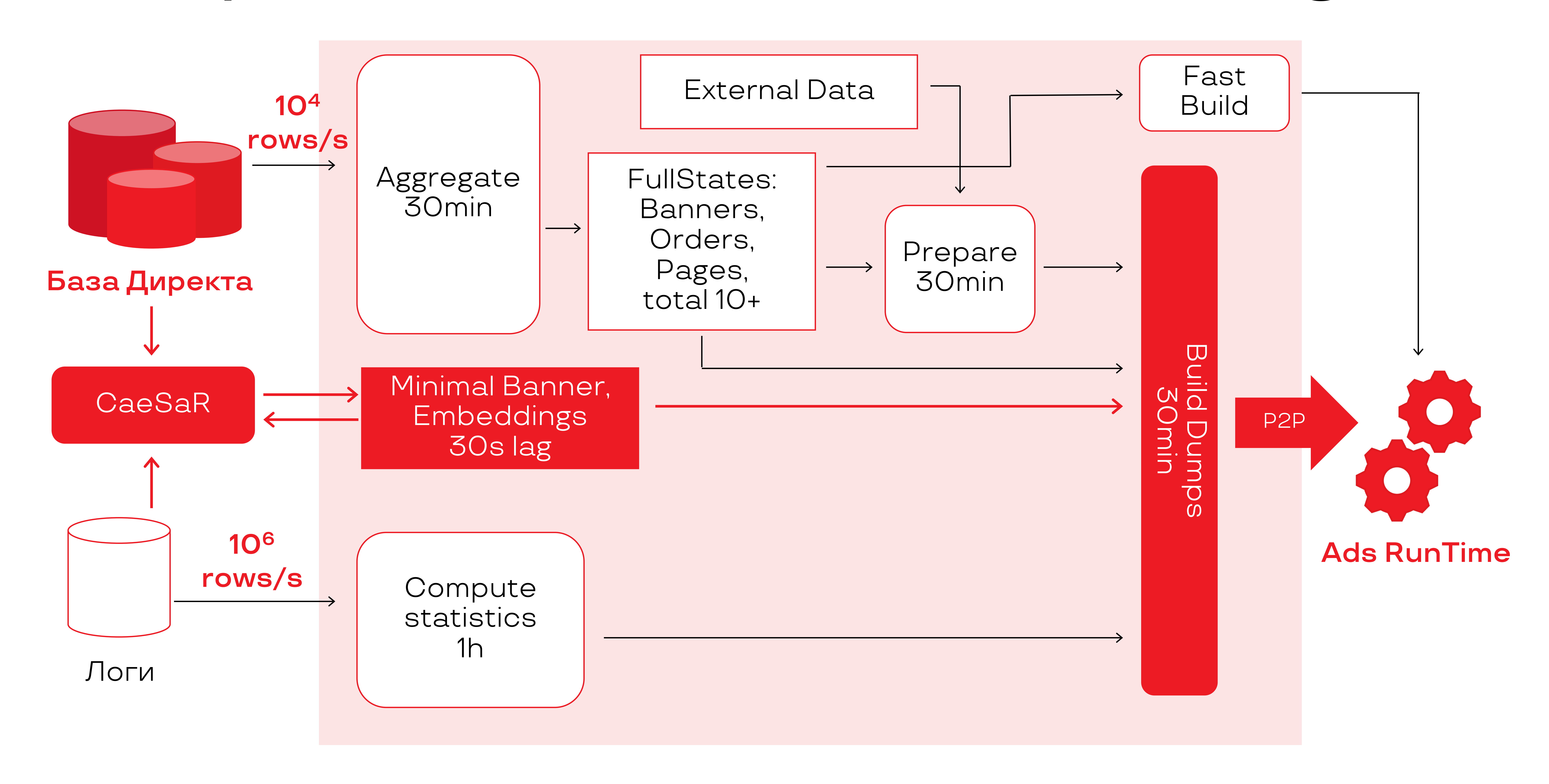
Доставка Banner Title – 2h



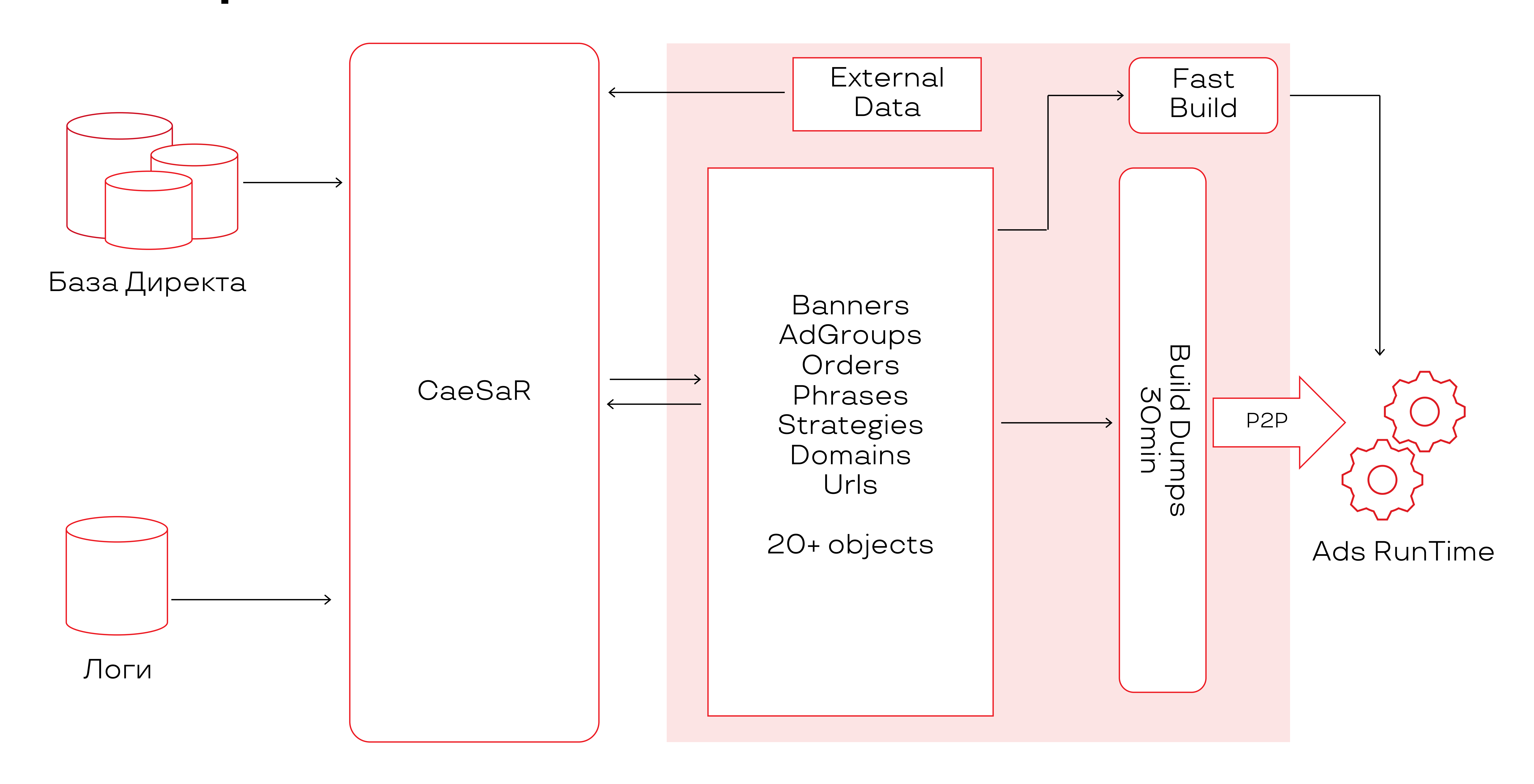
Доставка Banner Embedding – 6h



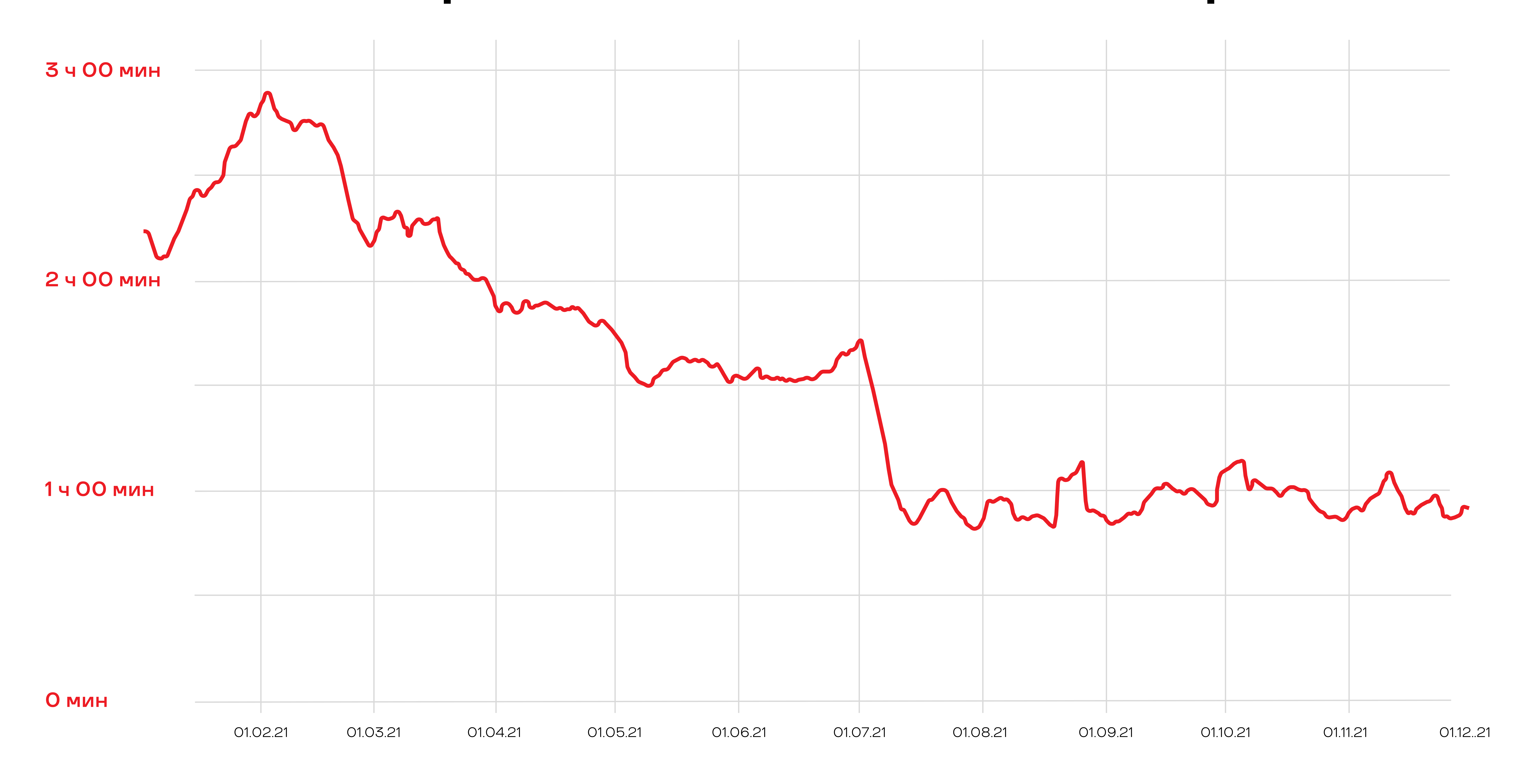
Внедрили MVP CaeSaR для Embeddings: 1h



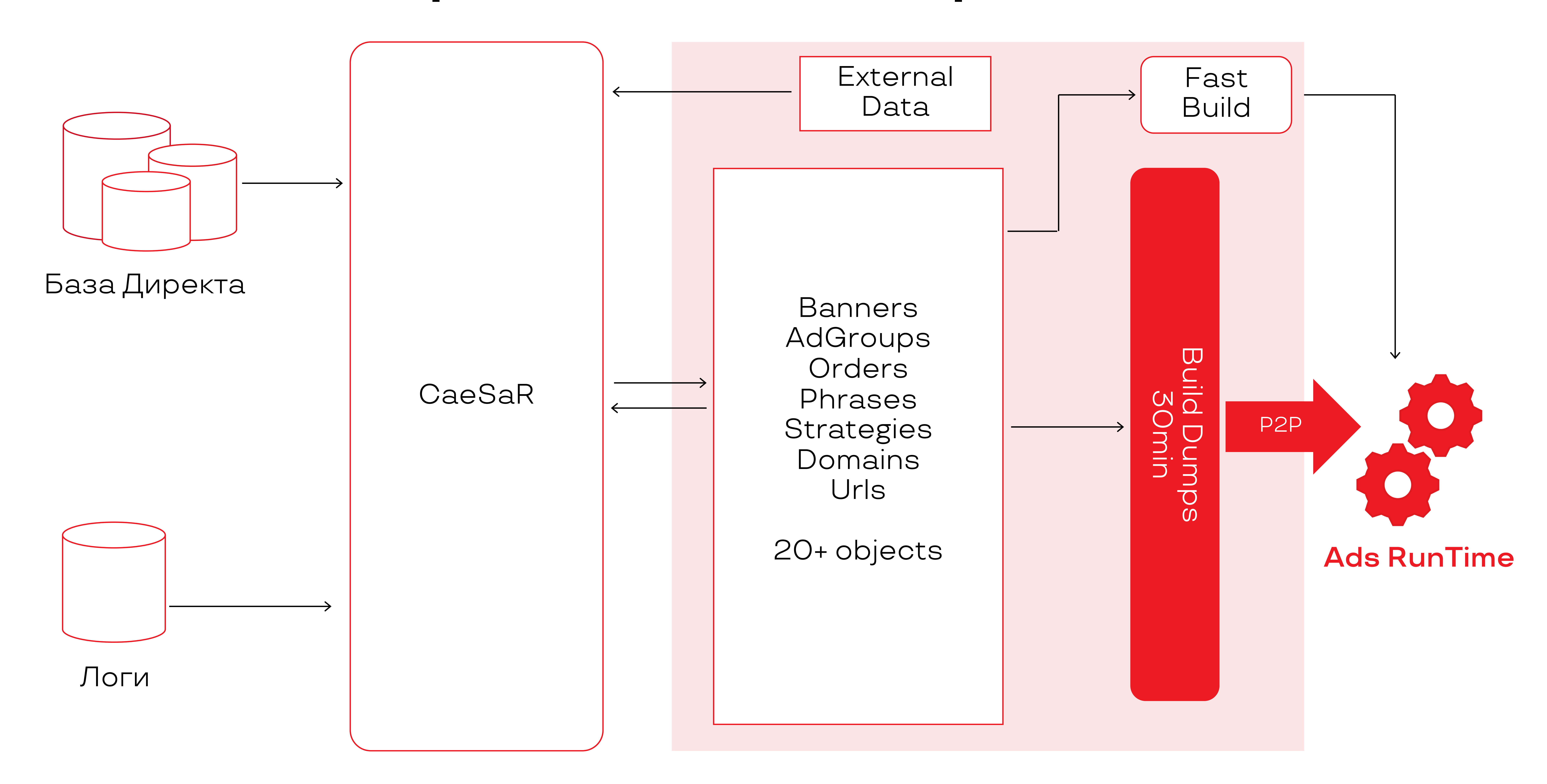
Внедрили CaeSaR для всех объектов



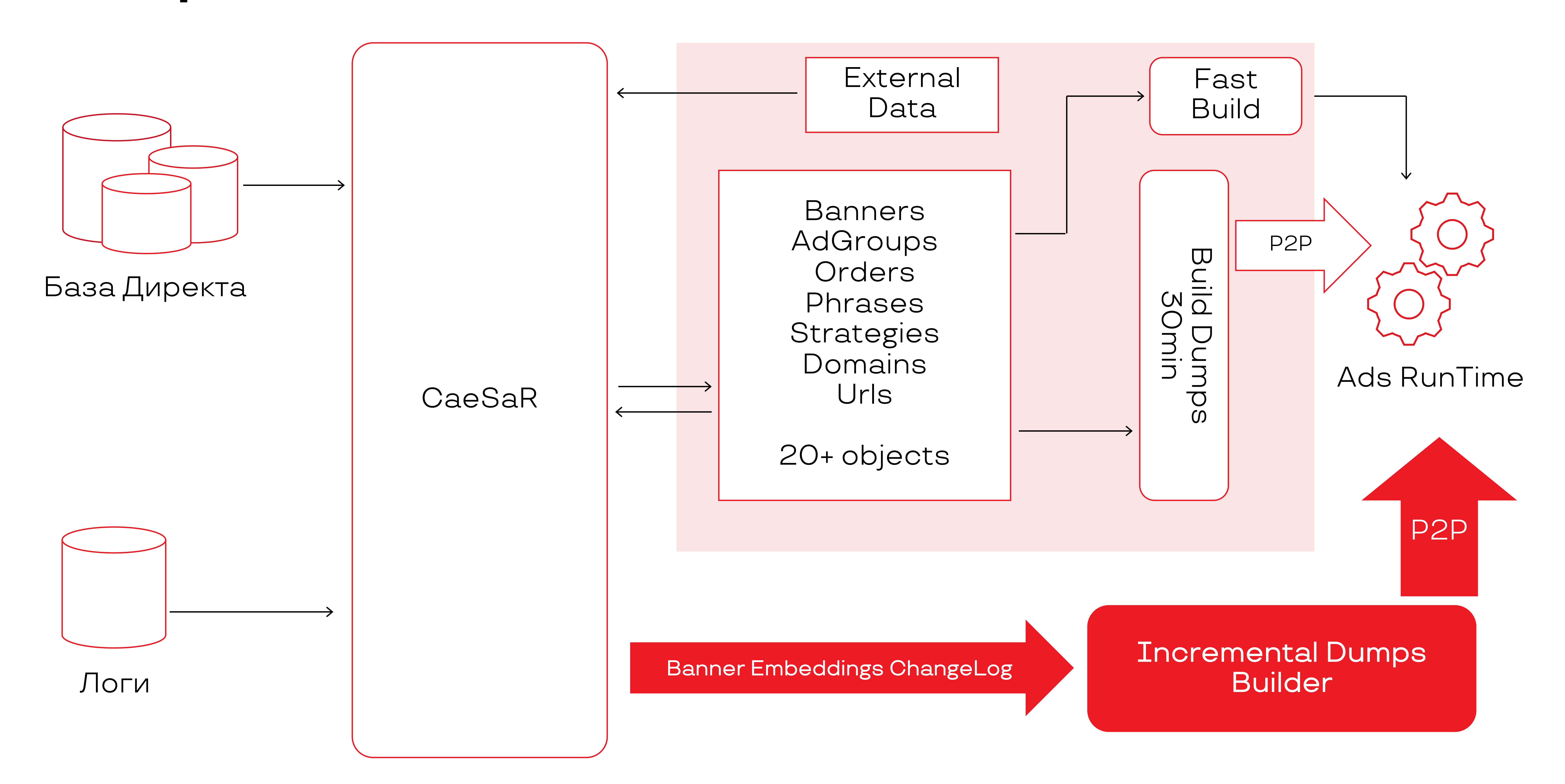
CaeSaR сократил лаг доставки в 2 раза



Осталась проблема в построении дампов



Инкрементальная доставка – MVP, 15min



Событийная архитектура – это хорошо даже в очень больших контент-системах

+ Можно достичь минутных и секундных лагов
+ Нет компромиссов между скоростью, объемом и полнотой
+ Легко масштабировать горизонтально
+ Упрощение разработки продукта

И о подводных камнях

- Смена парадигмы – нужно обучать разработчиков
- Массивное переписывание бизнес-логики
- Трансформация занимает время (у нас – годы), сложно ускорить
- Нужно серьезно подойти к выбору базовой инфраструктуры: DB, CI/CD
- Переосмысление систем восстановления

+ Нужно продукту
+ Нужно инфраструктуре

Будущее за событийной архитектурой

- Нужно продукту
- Нужно инфраструктуре

Спасибо!

Егор Хайруллин
руководитель отдела
инфраструктуры рекомендательных систем

mikari@yandex-team.ru

@mikari



Яндекс